Aalto University
School of Science

Benchmarking, Monitoring, Observability, and Experimenting for Big Data and Machine Learning Systems

Hong-Linh Truong
Department of Computer Science
linh.truong@aalto.fi, *https://rdsea.github.io*

CS-E4660 Advanced Topics in Software Systems, Fall 2023
13/09/2023

Learning objectives

- Able to analyze the role of measurement, monitoring and observability in real-world cases for R3E

- Understand and develop methods with key steps and important tools for benchmarking, monitoring, observability and experimenting

- Able to apply these methods for big data/ML systems

The role of measurement, monitoring and observability

Development vs Runtime activities

Design, test and benchmark R3E

- R3E for individual components
- model/capture complex dependencies
- design logs, metrics and traces for capturing states and complex dependencies

Monitoring/observability and runtime adaptation

- runtime monitoring and observability
- states, performance and failure analytics
- runtime controls (constraints, rules, actions)

Measurement, monitoring, and observability for R3E

- **Instrumentation and sampling**
  - instrumentation: insert probes into systems to measure system behaviors directly or produce logs
  - sampling: use components to sample system behaviors
- **Monitoring**
  - perform sampling or instrumentation to collect and share metrics, logs, traces; visualize what has been happened
- **Observability**
  - evaluate and interpret measurements for specific contexts
  - understand and explain the systems states, dependencies, etc.

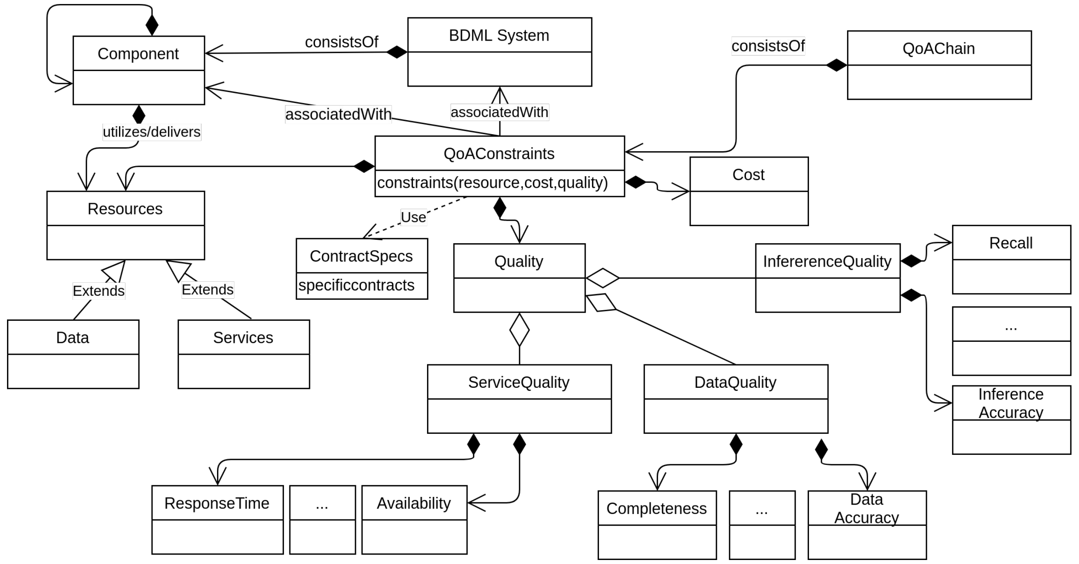
Recall: strongly interdependencies

Any problem would lead to a huge waste (engineering effort, operation cost, societal impact due to wrong inference/prediction)

as a process
(e.g. feature engineering, training process)

as an artefact
(e.g. trained model)

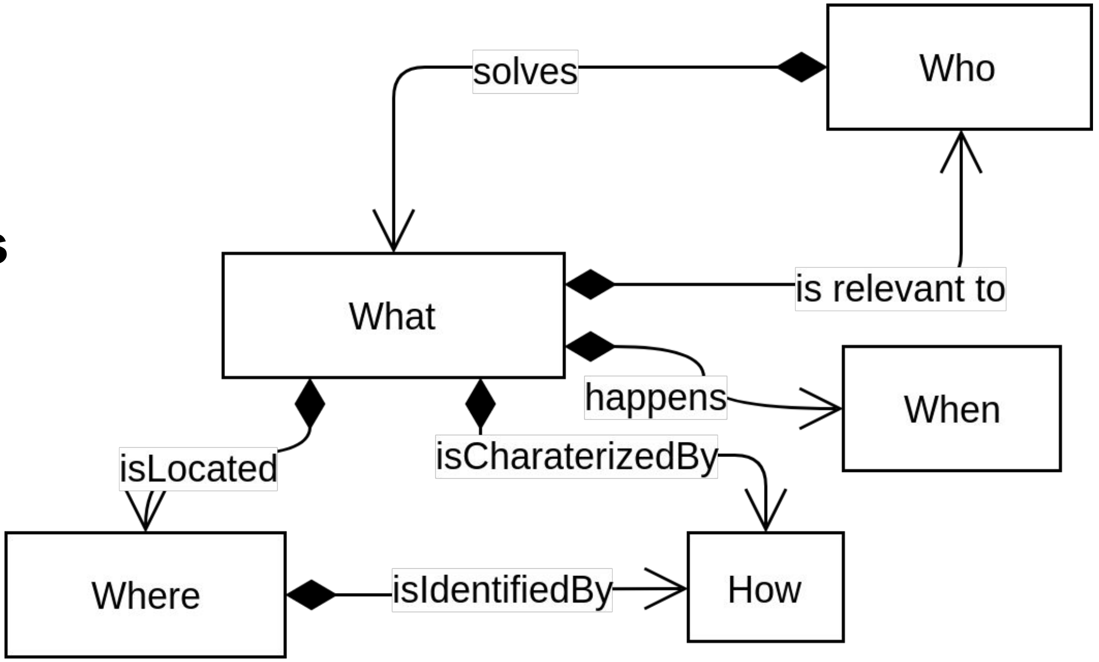
as a service
(e.g., a deployed ML prediction)



Methods

What/Which, Where, When, Who and How

Understand W4H aspects for analytics of big data/ML systems



Key steps – What/Which

- **Understand and identify indicators/metrics characterizing your systems**
- **Common metrics vs specific (big data/ML) ones**
  - different relevance/importance based on specific contexts
- **Most critical problems are due to complex dependencies that are not common**
  - root cause analysis will be tricky
- **For which purposes?**
  - SRE, benchmarking, Test-Driven Development (TDD)

Key steps – Where and When

- **Where: as a “ space” dimension**
  - tightly coupled or isolated/loosely coupled
  - different places
    - software/system layers, components and systems boundaries
    - dependencies among components
    - development/configuration pipelines
- **When: as a „time“ dimension**
  - design, test/training, or runtime (DevOps)
  - further divided into sub states

Key steps - How

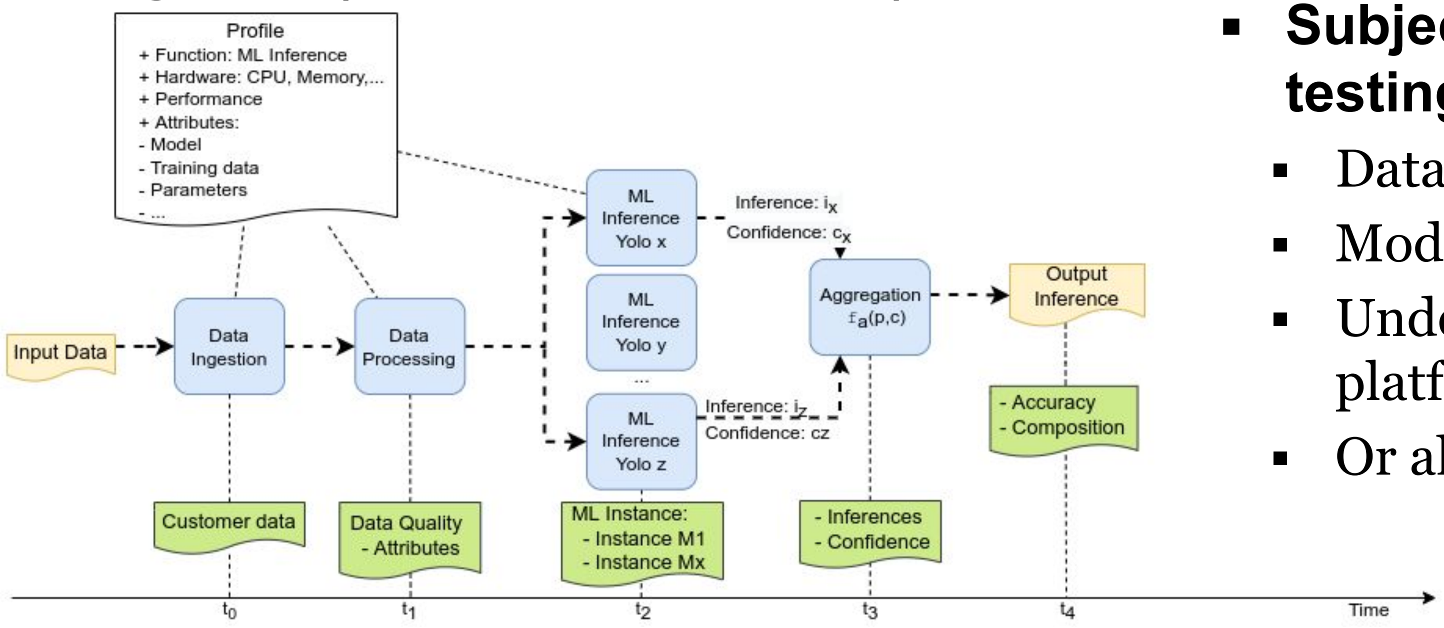
- **Characterize dependencies among components**
    - understand the system as a whole
    - include also development processes, data, software artefacts and execution environments
- **Select tools for capturing metrics**
- **Understand what kind of changes/designs we must do**
- **Do monitoring and analysis**
- **Integrate many types of data for monitoring and observability**

Apply W4H for benchmarking, monitoring, validation and experimenting

- **Determines clearly system boundaries**
  - the system under study, the system used to judge, and the environment
  - “domain-driven/oriented” and bounded context principles
- **Understands dependencies**
  - among components in distributed big data/ML systems in distributed computing platforms
  - single layer as well as cross-layered dependencies
- **Determines types of metrics and failures and break down problems along the dependency path (how)**

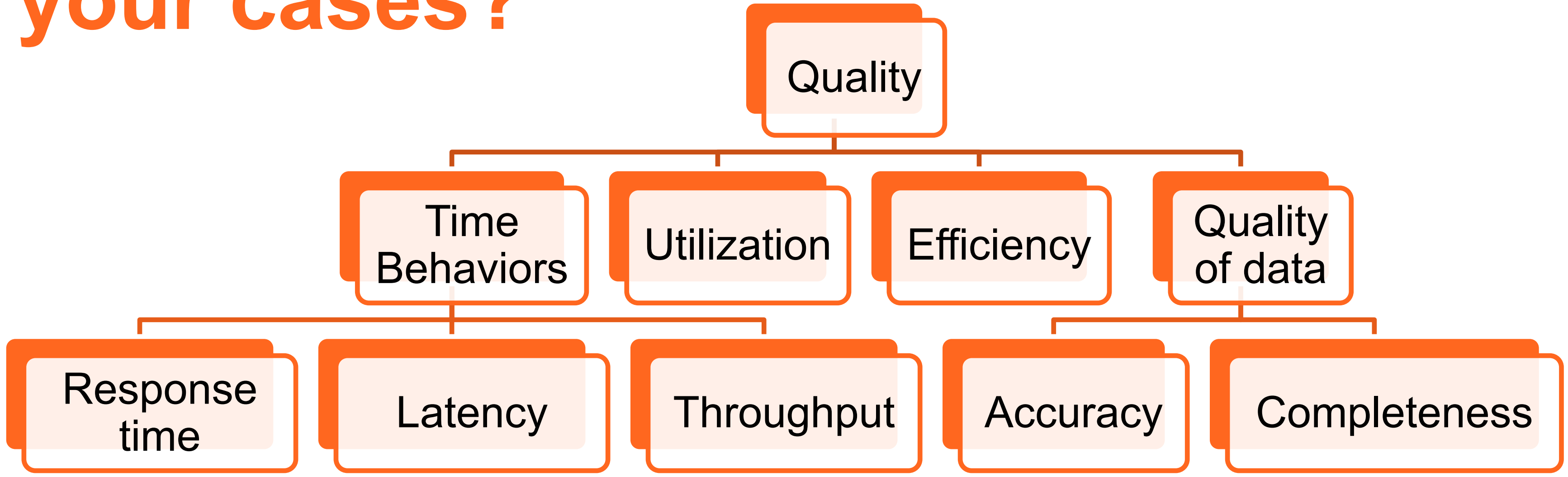
Boundaries and dependencies

Example of a ML service for object recognition (used in our hands-on)



- **Subjects for testing/debugging**
  - Data?
  - Model?
  - Underlying service platform?
  - Or all of them?

What are the most critical metrics for your cases?



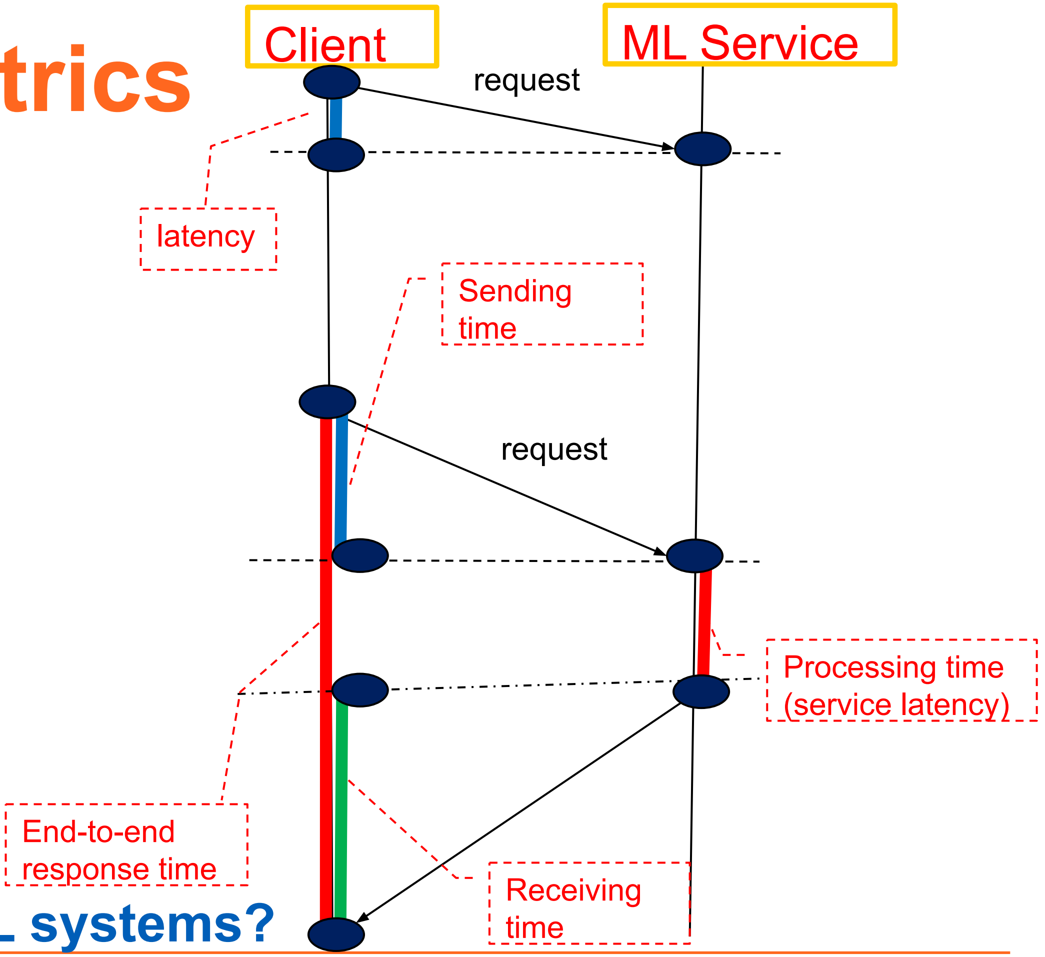
Industry view: https://guidingmetrics.com/content/cloud-services-industrys-10-most-critical-metrics/
NIST: https://www.nist.gov/sites/default/files/documents/itl/cloud/RATAX-CloudServiceMetricsDescription-DRAFT-20141111.pdf

Contradiction/Tradeoffs between Efficiency versus Resiliency
Metrics for an ML model =! Metrics for ML system

Common performance metrics

- **Timing behaviors**
  - Communication
    - *Latency/Transfer time*
    - *Data transfer rate, bandwidth*
  - Processing
    - *Response time (service latency/time)*
    - *Throughput*
- **Utilization**
  - Network utilization
  - CPU utilization
  - Service utilization
- **Efficiency/Scalability**
  - Concurrent executions

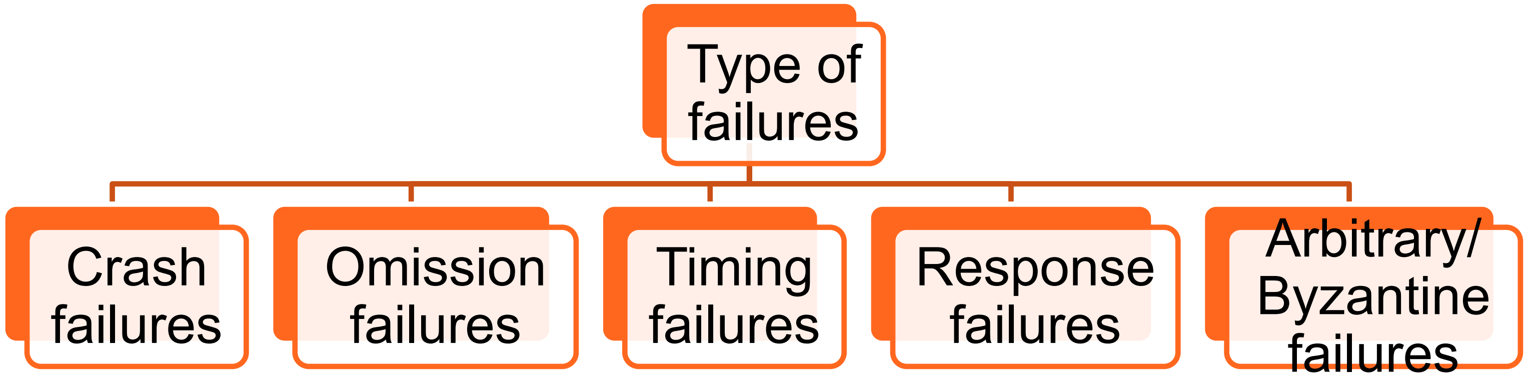
Examples



are they enough for big data/ML systems?

Types of Failure

Common



But unforeseen failures cannot be determined in advance design for handling failure

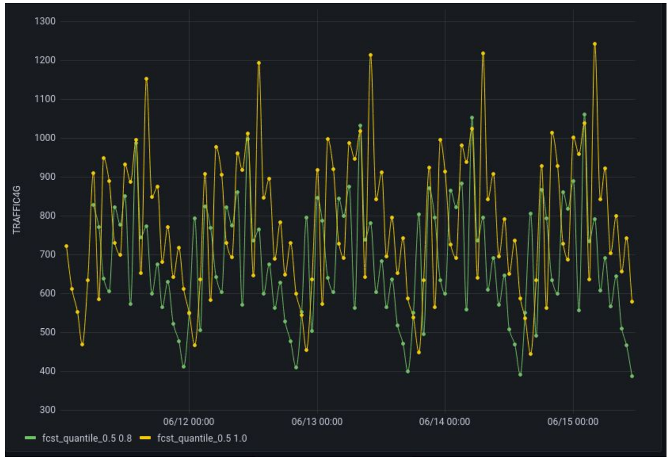
Check: https://arxiv.org/pdf/1910.11015.pdf for a “Taxonomy of Real Faults in Deep Learning Systems”

Metrics for Data

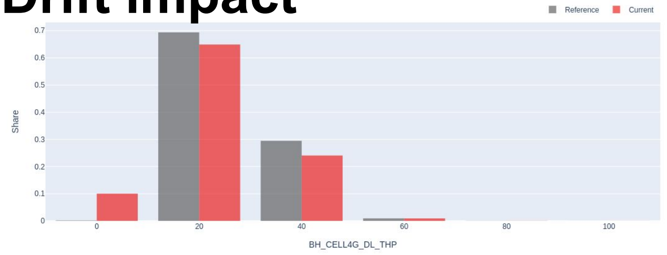
- Completeness
- Timeliness
- Currency
- Validity
- Format
- Accuracy
- Data Drift

Understand the impact

Forecasting



Drift impact



Often evaluation methods are different for different types of data

(examples with real mobile data)

Metrics for ML models

- Confusion matrix
- Accuracy
- Loss
- True positive rate
- False positive rate
- F1 Score/F-measure
- Etc.

(see https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234)

How would we define "reliable function" of the model? E.g., when should we "retrain" the model?

But how can we relate them to ML service/system?

Aalto University
School of Science

Benchmarking and Observability

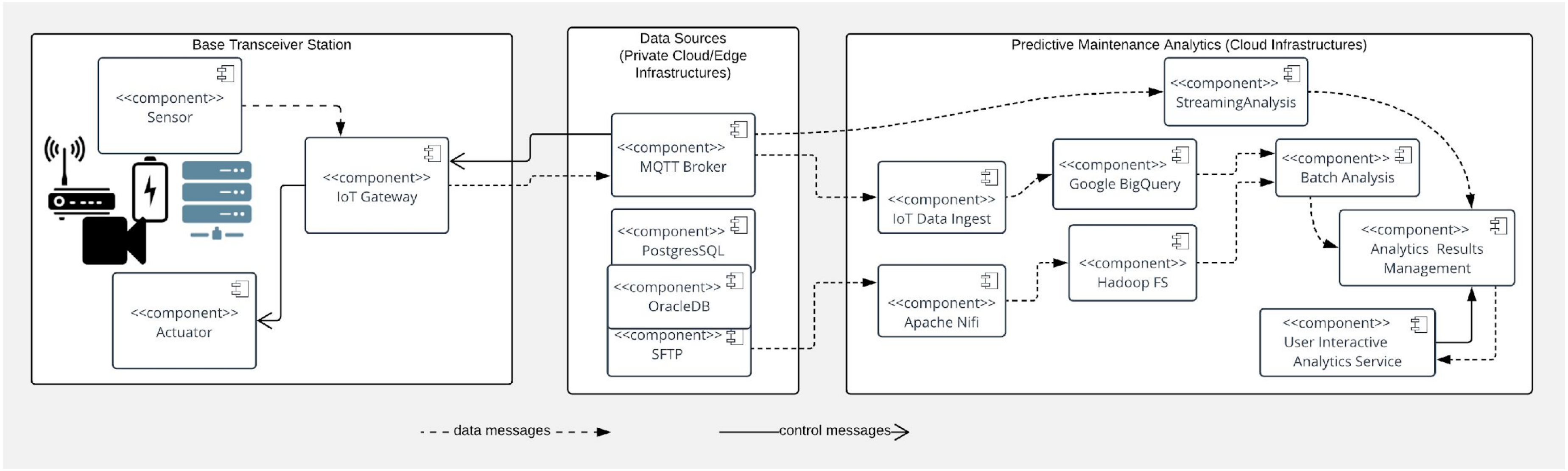
CS-E4660 Advanced Topics in Software Systems, Fall 2023
13/09/2023

Benchmarking

- **Benchmark: for comparing big data/ML systems w.r.t. selected (standard/common) workloads**
- **Where to be benchmarked**
  - benchmark individual subsystems: message brokers and data ingestion, databases and ingestion/query, data processing, ML models, serving platform
- **What to be benchmarked**
  - data ingestion throughput, processing throughput and time, component CPU and memory
  - training and inferencing time and accuracy

Benchmarking

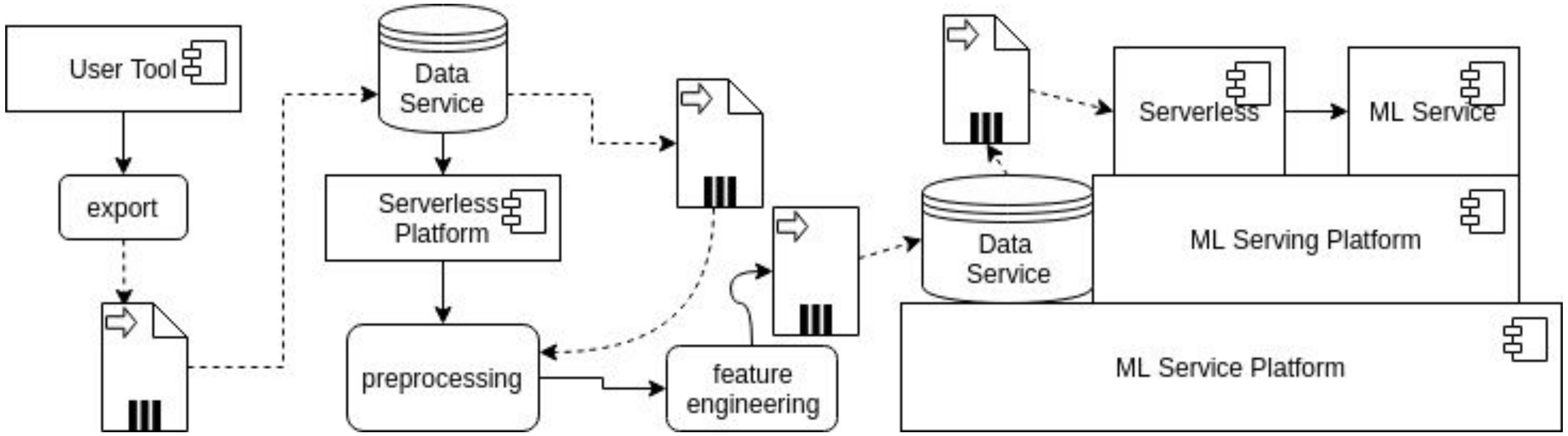
What should we do for a big data system?


Base Transceiver Station
<<component>> Sensor
<<component>> IoT Gateway
<<component>> Actuator
Data Sources (Private Cloud/Edge Infrastructures)
<<component>> MQTT Broker
<<component>> PostgresSQL
<<component>> OracleDB
<<component>> SFTP
Predictive Maintenance Analytics (Cloud Infrastructures)
<<component>> StreamingAnalysis
<<component>> IoT Data Ingest
<<component>> Google BigQuery
<<component>> Batch Analysis
<<component>> Analytics Results Management
<<component>> Apache Nifi
<<component>> Hadoop FS
<<component>> User Interactive Analytics Service
data messages
control messages

Check:
https://www.sciencedirect.com/science/article/pii/S0140366419312344
https://www.benchcouncil.org/BigDataBench/

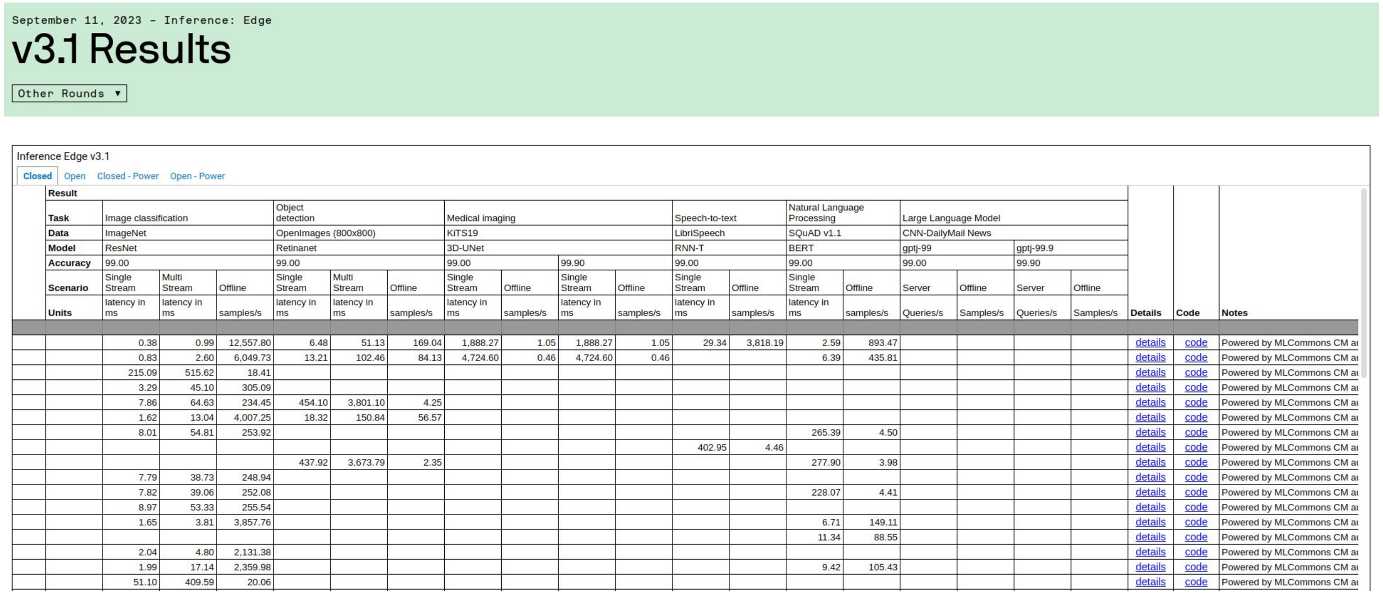
Benchmarking

If you have an end-to-end ML system, does it make sense to benchmark the whole system?



Benchmarking - ML

Examples:



September 11, 2023 - Inference: Edge

v3.1 Results

Other Rounds ▾

Inference Edge v3.1

Closed | Open | Closed - Power | Open - Power

Result																					
Task	Image classification			Object detection			Medical imaging				Speech-to-text		Natural Language Processing		Large Language Model						
Data	ImageNet			OpenImages (800x800)			KiTS19				LibriSpeech		SQuAD v1.1		CNN-DailyMail News						
Model	ResNet			Retinanet			3D-UNet				RNN-T		BERT		gptj-99		gptj-99.9				
Accuracy	99.00			99.00			99.00		99.90		99.00		99.00		99.00		99.90				
Scenario	Single Stream	Multi Stream	Offline	Single Stream	Multi Stream	Offline	Single Stream	Offline	Single Stream	Offline	Single Stream	Offline	Single Stream	Offline	Server	Offline	Server	Offline			
Units	latency in ms	latency in ms	samples/s	latency in ms	latency in ms	samples/s	latency in ms	samples/s	latency in ms	samples/s	latency in ms	samples/s	latency in ms	samples/s	Queries/s	Samples/s	Queries/s	Samples/s	**Details**	**Code**	**Notes**
	0.38	0.99	12,557.80	6.48	51.13	169.04	1,888.27	1.05	1,888.27	1.05	29.34	3,818.19	2.59	893.47					details	code	Powered by MLCommons CM a
	0.83	2.60	6,049.73	13.21	102.46	84.13	4,724.60	0.46	4,724.60	0.46			6.39	435.81					details	code	Powered by MLCommons CM a
	215.09	515.62	18.41																details	code	Powered by MLCommons CM a
	3.29	45.10	305.09																details	code	Powered by MLCommons CM a
	7.86	64.63	234.45	454.10	3,801.10	4.25													details	code	Powered by MLCommons CM a
	1.62	13.04	4,007.25	18.32	150.84	56.57													details	code	Powered by MLCommons CM a
	8.01	54.81	253.92										265.39	4.50					details	code	Powered by MLCommons CM a
											402.95	4.46							details	code	Powered by MLCommons CM a
				437.92	3,673.79	2.35							277.90	3.98					details	code	Powered by MLCommons CM a
	7.79	38.73	248.94																details	code	Powered by MLCommons CM a
	7.82	39.06	252.08										228.07	4.41					details	code	Powered by MLCommons CM a
	8.97	53.33	255.54																details	code	Powered by MLCommons CM a
	1.65	3.81	3,857.76										6.71	149.11					details	code	Powered by MLCommons CM a
													11.34	88.55					details	code	Powered by MLCommons CM a
	2.04	4.80	2,131.38																details	code	Powered by MLCommons CM a
	1.99	17.14	2,359.98										9.42	105.43					details	code	Powered by MLCommons CM a
	51.10	409.59	20.06																details	code	Powered by MLCommons CM a

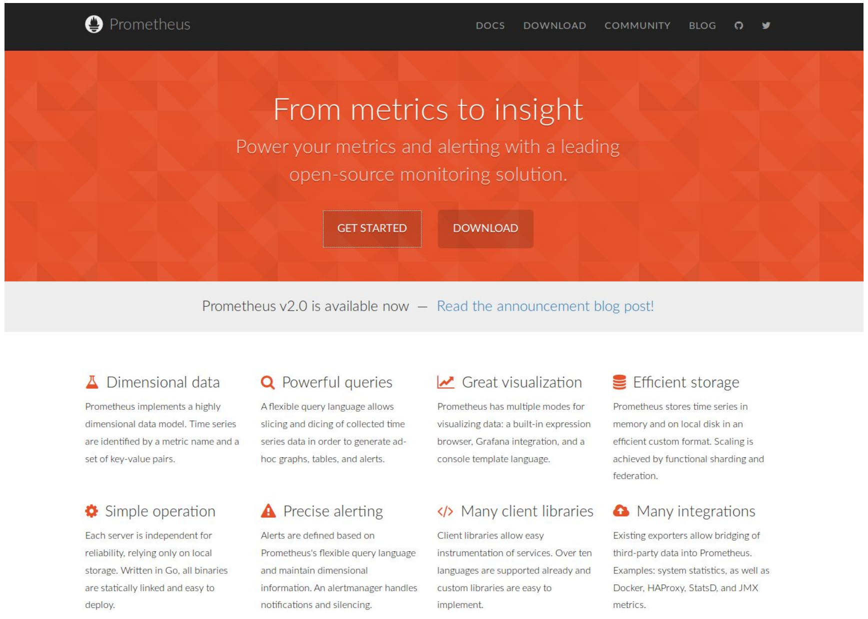
Source: https://mlcommons.org/en/inference-edge-31/

Also check: https://www.benchcouncil.org/aibench/index.html

Service/Infrastructure monitoring tools

There are many powerful tools!

But only low-level, well-identified monitoring data (infrastructures): pre-defined metrics exposed through interfaces with push/pull mechanism



From metrics to insight

Power your metrics and alerting with a leading open-source monitoring solution.

Prometheus v2.0 is available now — Read the announcement blog post!

Dimensional data
Prometheus implements a highly dimensional data model. Time series are identified by a metric name and a set of key-value pairs.

Powerful queries
A flexible query language allows slicing and dicing of collected time series data in order to generate ad-hoc graphs, tables, and alerts.

Great visualization
Prometheus has multiple modes for visualizing data: a built-in expression browser, Grafana integration, and a console template language.

Efficient storage
Prometheus stores time series in memory and on local disk in an efficient custom format. Scaling is achieved by functional sharding and federation.

Simple operation
Each server is independent for reliability, relying only on local storage. Written in Go, all binaries are statically linked and easy to deploy.

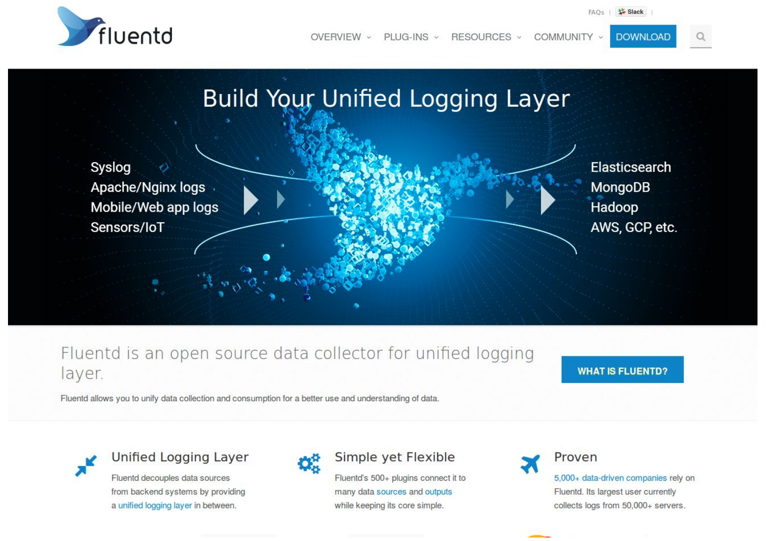
Precise alerting
Alerts are defined based on Prometheus's flexible query language and maintain dimensional information. An alertmanager handles notifications and silencing.

Many client libraries
Client libraries allow easy instrumentation of services. Over ten languages are supported already and custom libraries are easy to implement.

Many integrations
Existing exporters allow bridging of third-party data into Prometheus. Examples: system statistics, as well as Docker, HAProxy, StatsD, and JMX metrics.

From: https://prometheus.io/

Instrumentation for observability



Build Your Unified Logging Layer

Syslog
Apache/Nginx logs
Mobile/Web app logs
Sensors/IoT

Elasticsearch
MongoDB
Hadoop
AWS, GCP, etc.

Fluentd is an open source data collector for unified logging layer.

WHAT IS FLUENTD?

Fluentd allows you to unify data collection and consumption for a better use and understanding of data.

Unified Logging Layer

Fluentd decouples data sources from backend systems by providing a unified logging layer in between.

Simple yet Flexible

Fluentd's 500+ plugins connect it to many data sources and outputs while keeping its core simple.

Proven

5,000+ data-driven companies rely on Fluentd. Its largest user currently collects logs from 50,000+ servers.

From: https://www.fluentd.org/

Code instrumentation: for many metrics and logs that cannot be obtained from the outside of the component

the developer can instrument the code to capture metrics/generate logs/traces

Filebeat

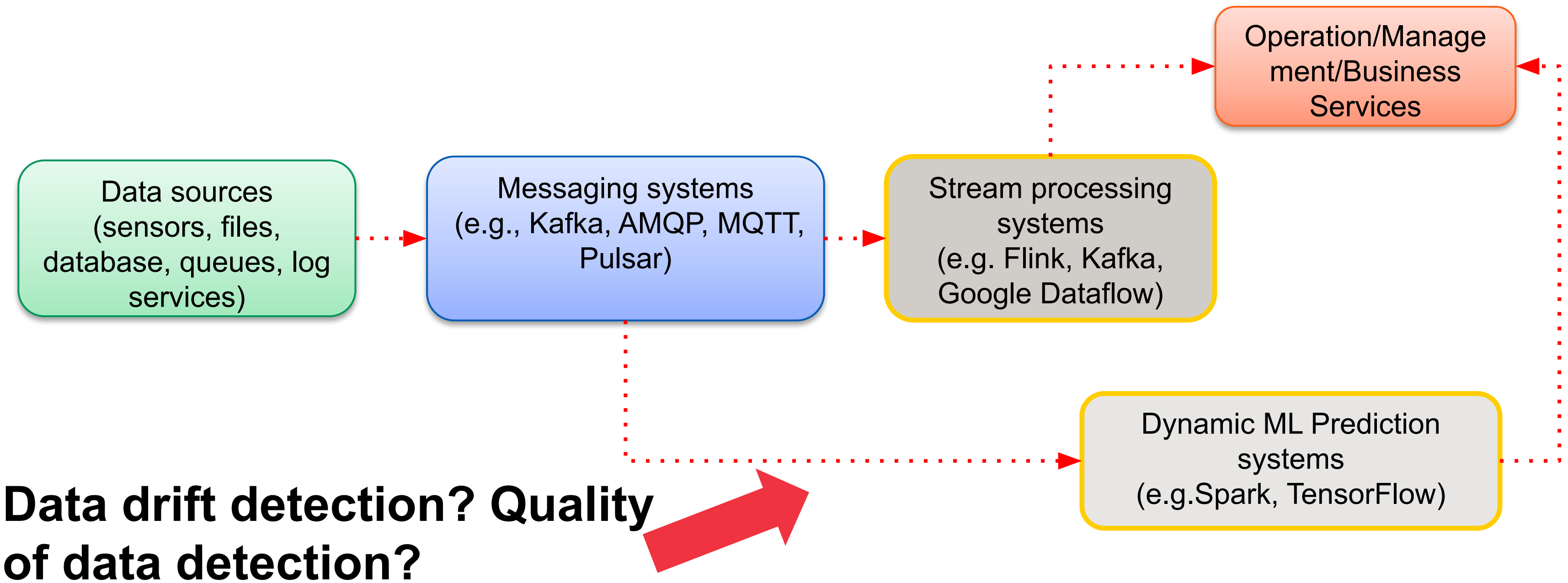
Lightweight shipper for logs

https://www.elastic.co/beats/filebeat

https://opentelemetry.io/

Aalto University
School of Science

Monitoring data metrics on-the-fly


Data sources (sensors, files, database, queues, log services)
Messaging systems (e.g., Kafka, AMQP, MQTT, Pulsar)
Stream processing systems (e.g. Flink, Kafka, Google Dataflow)
Operation/Management/Business Services
Dynamic ML Prediction systems (e.g.Spark, TensorFlow)

Data drift detection? Quality of data detection?
Or performance prediction?

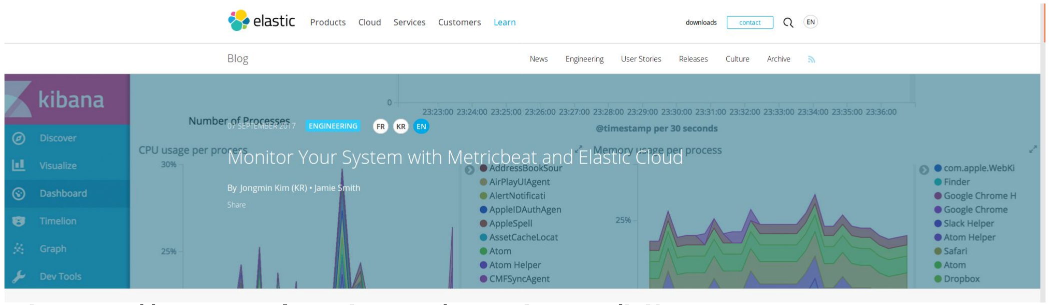
A couple of tools for data quality

- **Generic tools/framework for checking data at rest**
  - Great expectation: https://github.com/great-expectations/great_expectations
  - YData (https://github.com/ydataai/ydata-quality)
  - Alibi-Detect (https://github.com/SeldonIO/alibi-detect)
  - Why-log (https://docs.whylabs.ai/docs/whylogs-overview/)
- **Integrated with processes in specific systems**
  - https://aws.amazon.com/blogs/industries/how-to-architect-data-quality-on-the-aws-cloud/
- **Working with specific data processing frameworks**
  - https://github.com/awslabs/python-deequ

Visualization

Metrics and Visualization

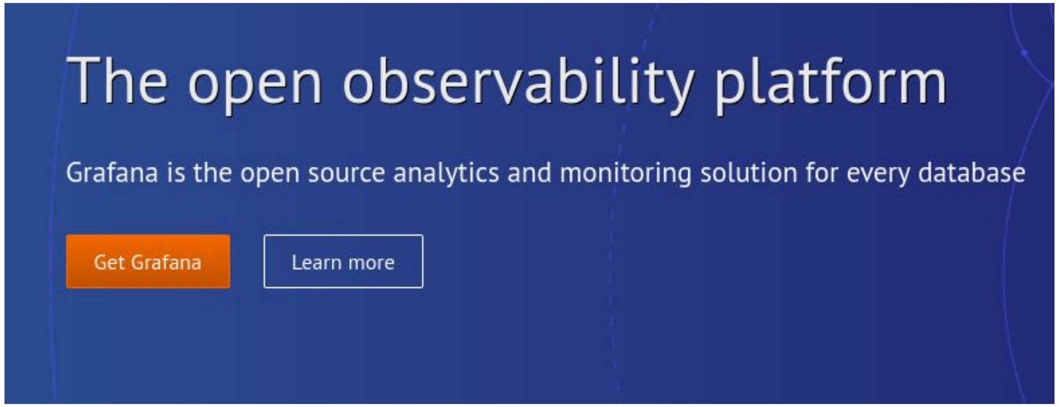
- **Easy to visualize many types of metrics**
  - Human-in-the-loop
- **But only you can specify, define and map them to your structured applications**
- **Not for complex process automation!**
  - further integration and intelligence analytics (ML?)



Monitor Your System with Metricbeat and Elastic Cloud

By Jongmin Kim (KR) • Jamie Smith

https://www.elastic.co/products/kibana



The open observability platform

Grafana is the open source analytics and monitoring solution for every database

https://grafana.com/

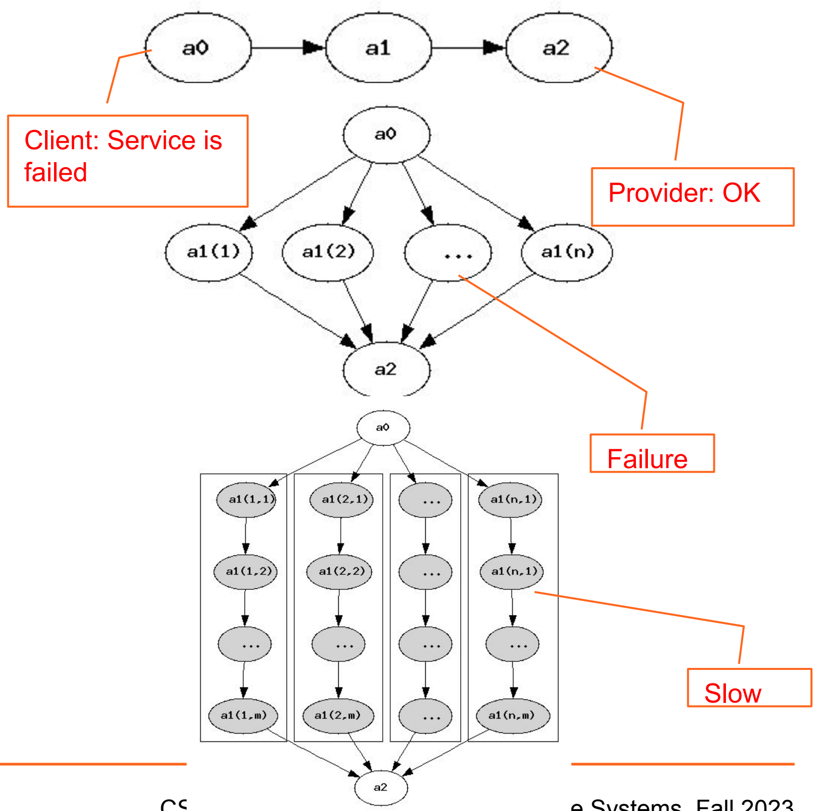
Observability

- **To monitor and understand the system as whole, end-to-end**
  - every component must be monitored
  - dependencies/interactions must be captured
  - diverse metrics, logs, tracing, etc. are needed to be integrated
- **Understand the states and behaviors of the whole systems**
- **Complex problems in big data/ML systems as these systems**
  - large-scale number of microservices in large-scale virtualized infrastructures
  - multi-dimensional states (code, models and data)

Understand the structure of big data/ML application

- **Composable method**
  - divide a complex structure into basic common structures
  - each basic structure has different ways to analyze specific failures/metrics
- **Interpretation based on context/view**
  - client view or service provider view?
  - conformity versus specific requirement assessment

Dependency Structure



Support an end-to-end view or not

- **End-to-end reflects the entire system**
  - e.g., data reliability: from sensors to the final analytics/inference results
  - what if the developer/provider cannot support end-to-end?
- **The user expects end-to-end R3E**
  - e.g., specified in the expected accuracy
- **Providers/operators want to guarantee end-to-end quality**
  - need to monitor different parts, each has subsystems/components
  - coordination-aware assurance, e.g., using elasticity

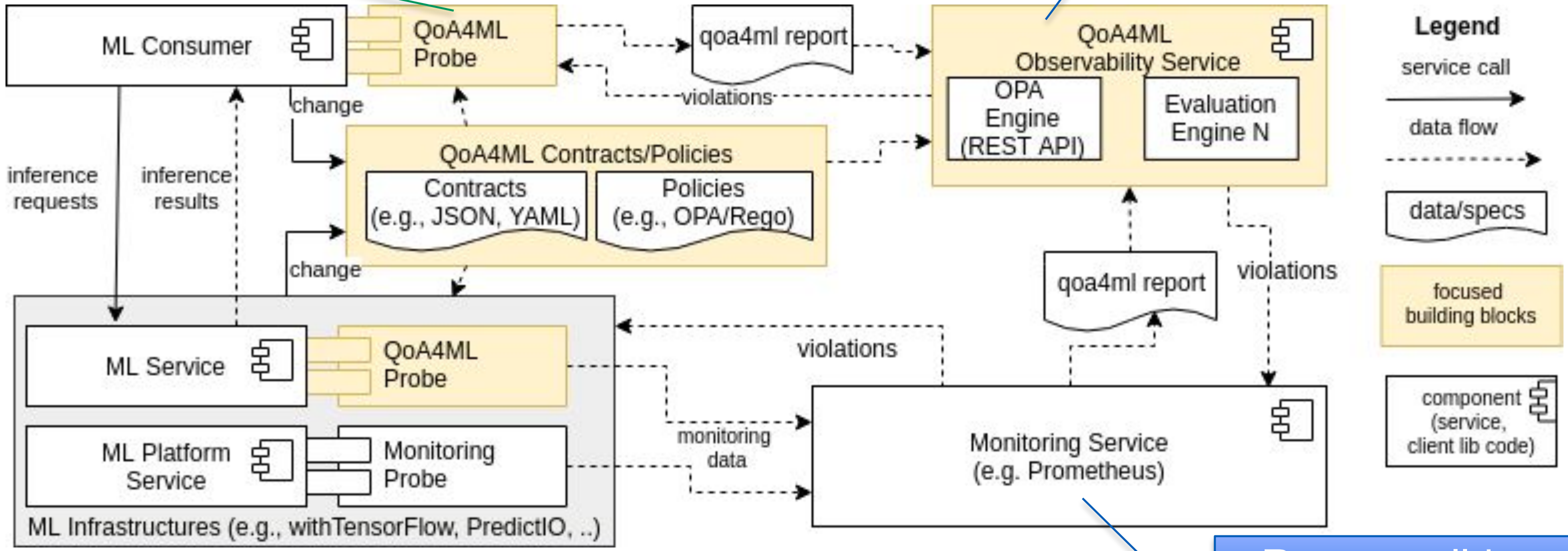
Big data/ML for Observability vs Observability for Big data/ML systems

- **Big data of metrics, logs and traces**
  - Large number of entities to be observed
  - High number of measurement dimensions
- **ML for observability**
  - Classification, prediction and detection of traffics/interactions anomaly behaviors, hidden relationships, etc.
  - Root-cause analysis
  - ML serving is   in the edge and cloud

Example: ML contract observability with QoA4ML

Monitor data/ML-specific attributes

Design for different engines to be used



Reuse well-known monitoring systems

https://github.com/rdsea/QoA4ML

Experiment management

how do we manage important information for ML services?

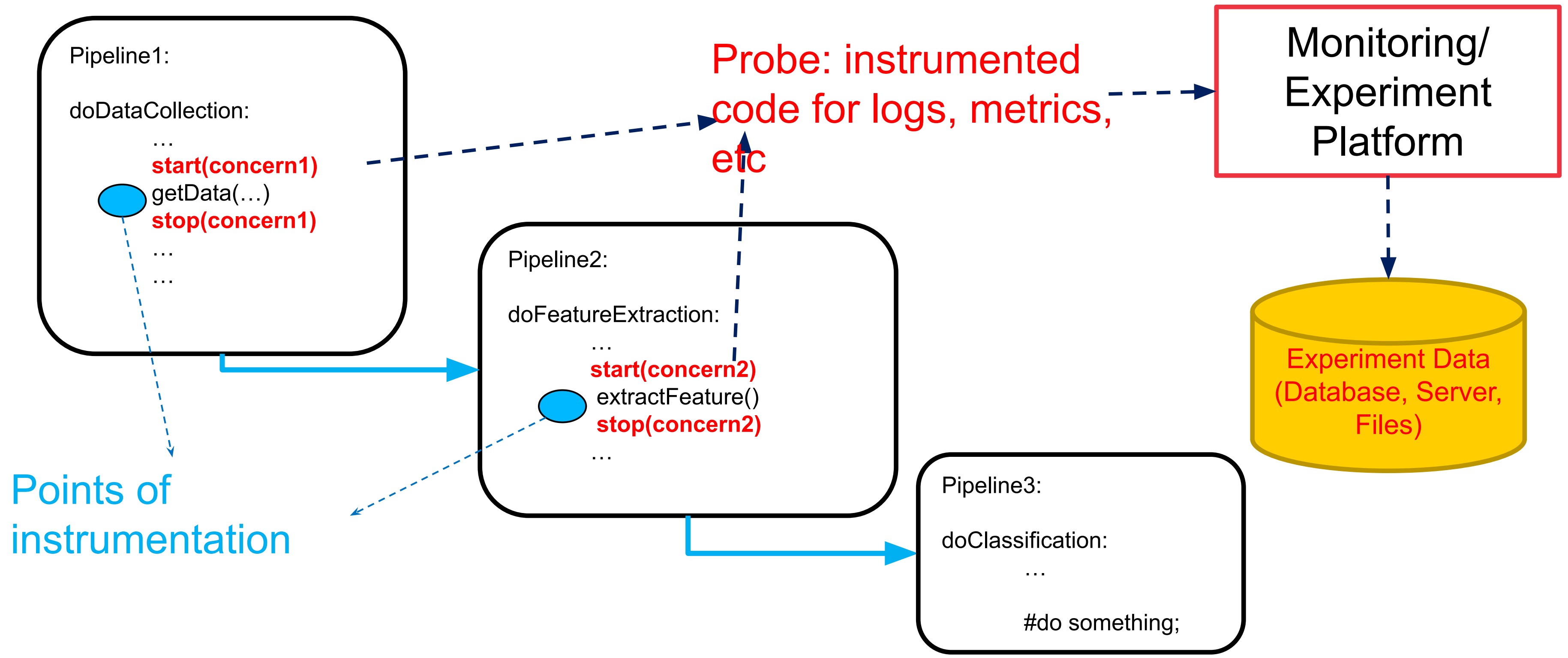
Problems

- **We need to run many experiments**
  - testability/observability purposes: figure out suitable configurations
  - how does this help to understand and support R3E?
- **Experiment management**
  - known domain and well-known books (e.g., "Design and Analysis of Experiments" by Douglas C. Montgomery)
  - principles: capturing various configurations
  - how does it work in big data and ML?
- **What do we need?**
  - tools/frameworks for tracking experiments

Notions

- **A single run/trial**
  - inputs, results, required software artefacts
  - computing resources, logs/metrics
- **Experiment**
  - a collection of runs/trials/executions gathered in a specific context
- **Steps**
  - parameterization: generate different parameters
  - deployment: prepare suitable environments
  - execution: run and collect metrics
  - analysis and sharing: analyze experiment data

Experiment tracking



But remember it is very large system! Different techniques/tools may be needed

Examples

- **Tensorflow Board (https://www.tensorflow.org/tensorboard)**
- **Experiment in Azure ML SDK**
  - https://docs.microsoft.com/en-us/python/api/overview/azure/ml/?view=azure-ml-py#experiment
- **MLFlows https://mlflow.org/**
- **Kubeflows**
  - https://www.kubeflow.org/docs/pipelines/overview/concepts/
- **DVC: https://dvc.org/**
- **Verta: https://www.verta.ai/**
- **Comet: https://www.comet.com/**

Examples: MLFlow APIs

- **Experiment**

  `mflow.start_run()/end_run()`

  `mflow.autolog()`

- **Logs/metrics collection**

  `mflow.set_tag()`

  `mflow.log_*()`

- **Tracking data management**
  - Local files, Databases, HTTP server, Databrick logs

(follow our hands-on tutorial)

Experiment management: more than just ML models

- **Remember there are many components in a system**
- **Experiment data about other components is also crucial**
  - have a full visibility and understanding of the system
  - support explainability and end-to-end optimization
- **ML model experiment must be combined with other types of experimental data**
  - experiment management for end-to-end systems

Study log 2

Describe one big data/ML pipeline that you are familiar with and explain your thoughts on how would you support the aspects of "benchmarking", "monitoring", "observability", or "experimenting" for testing/implementing R3E aspects

- Is enough to focus on 1 pipeline and 1 aspect
  - *No "familiar pipeline" → look at our hands-on tutorials*
- Be concrete, e.g., with metrics and possible tools
- Analyze if things can be done easily or where are the challenges that might be interesting for further investigation
- Optionally link to issues raised/addressed in a reading paper

Thanks!

Hong-Linh Truong
Department of Computer Science

rdsea.github.io